Reinforcement Learning for N-Player Games:

The Importance of Final Adaptation

Wolfgang Konen, Samineh Bagheri

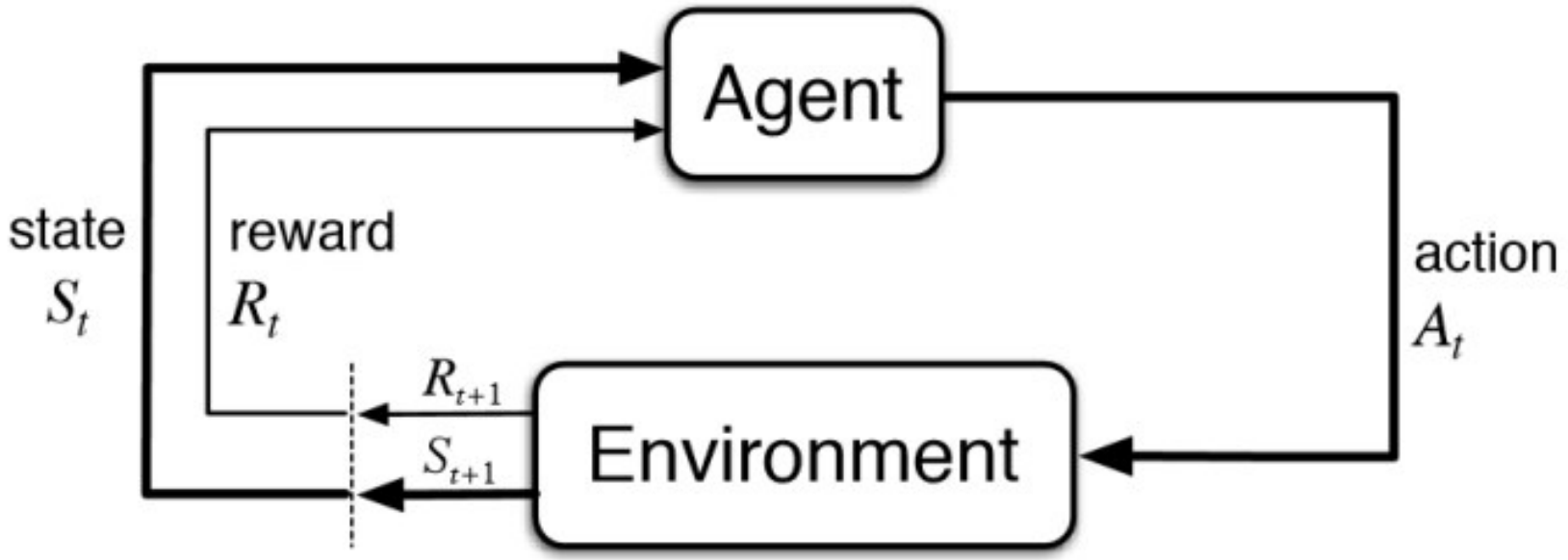
TH Köln, Cologne Institute of Computer Science

November 2020

Motivation and Contributions

- Motivation
  - Goal: Game learning from self-play, one algorithm for many different games
  - Game learning = complex (usually discrete and non-convex) **optimization** task
  - State spaces are often vast
  - → Use **R**einforcement **L**earning (RL) to explore and learn by self-play
- Contributions of this work
  - Unifying RL algorithm for games with $N = 1, 2, 3, ...$ players
  - A new element, called Final Adaptation RL (FARL), is vital to have success → we name our algorithm **TD-FARL**
  - TD-FARL combines several elements from advanced game learning (n-tuples, temporal coherence, afterstates and others)
  - Applied to seven quite different games: TD-FARL has higher win rates than other RL approaches

Reinforcement Learning



(c) Shweta Bhatt, www.kdnuggets.com

Temporal Difference (TD) Learning

- The agent follows a policy that maximizes the target

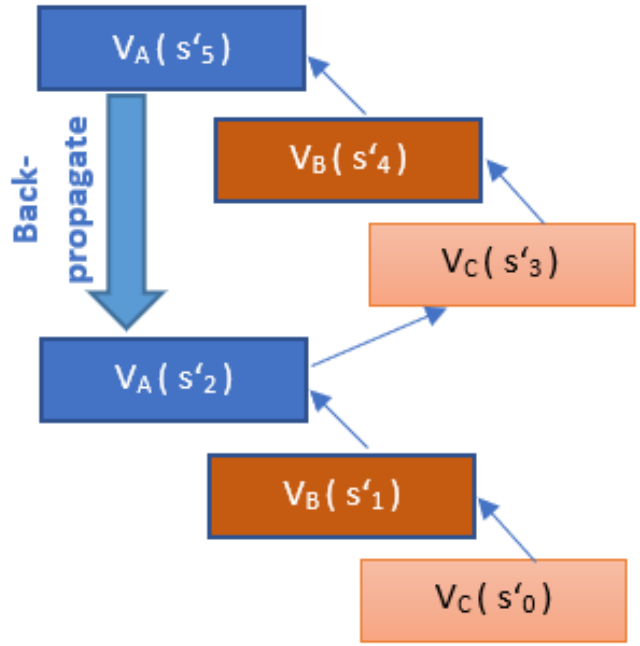
$$T = r_t + \gamma V(s'_t)$$

  where r_t and s'_t are reward and afterstate when taking action a_t in state s_t.
- Here, $V(s'_t)$ is the internal model of the agent: the *value* or expected sum of future rewards when being in s'_t.
- We use a model based on n-tuples.
- Learning: Backpropagate rewards + values to earlier states

$$V(s'_{t-1}) \leftarrow r_t + \gamma V(s'_t)$$

The Problem with N Players

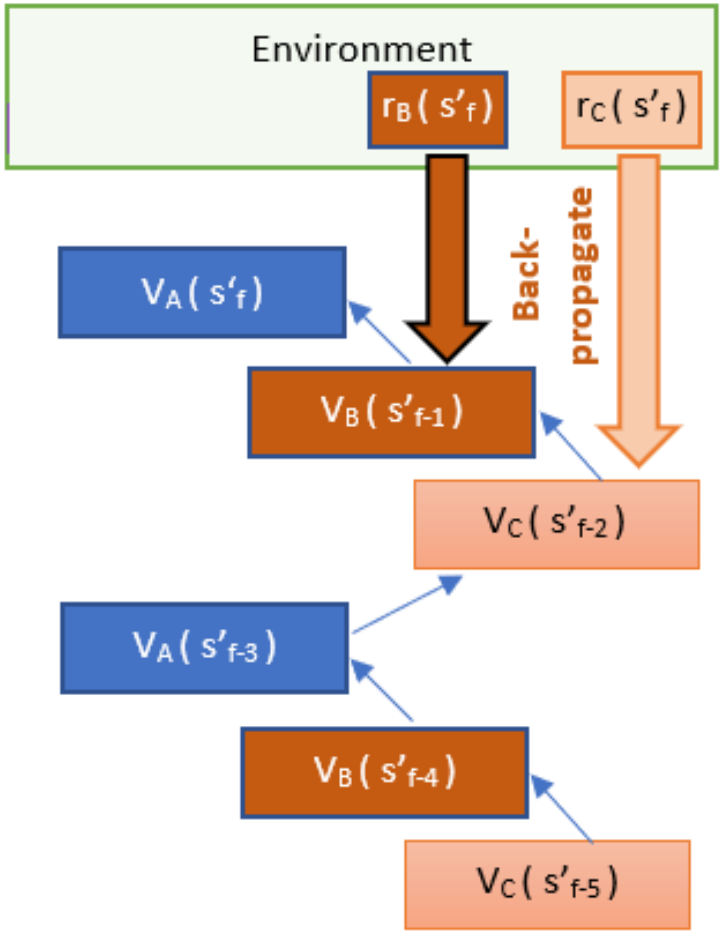
- In general, if A and B are two players, the relation between $V_A(s'_t)$ and $V_B(s'_t)$ is unknown.
- Therefore, backpropagation of values to *one move* earlier is difficult.
- Solution: backpropagate to ***one round* earlier** [van der Ree, 2013].
- Advantage: Works for $N = 1, 2, 3, \ldots$ in exactly the same way.



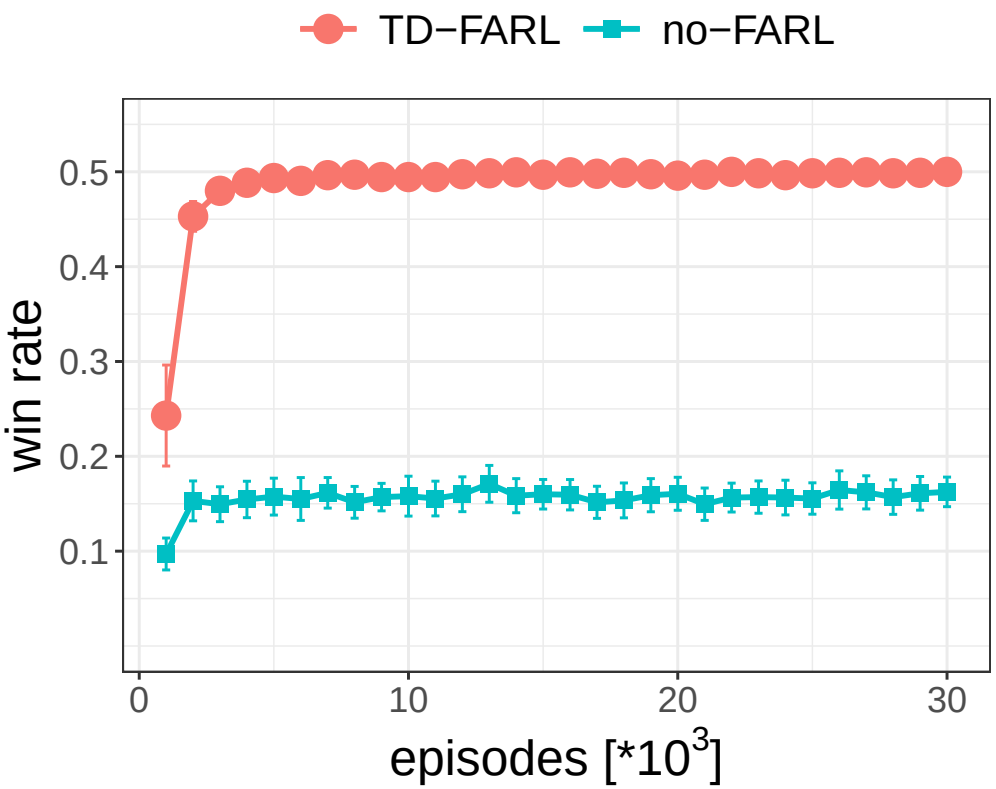
Backpropagate to *one round* earlier

Final Adaptation RL (FARL)

- Once we reach a final (game-over) state s'_f, the rewards for all players are known.
- Let's assume that player A wins with his final move and B and C lose.
- Normal RL: Only backpropagate reward r_A to last state of A (not shown).
- The **new ingredient of FARL**: Backpropagate for the other players (B and C) the (negative) reward to their last state.
- Thus, B and C learn that they left a threatening position to player A.



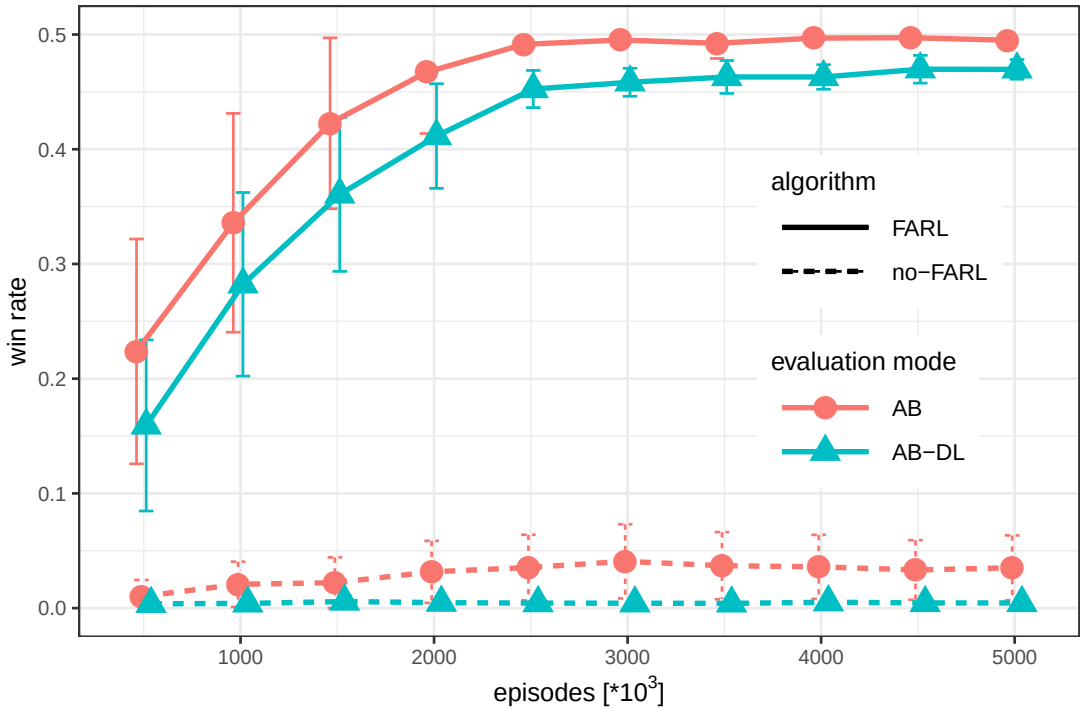
Results TicTacToe



TD-FARL on TicTacToe, 25 runs

- TicTacToe: TD-FARL vs. perfect-playing Max-N tree search agent.
- The ideal win rate against Max-N is 0.5.
- Drastic performance decrease when FARL is turned off.

Results ConnectFour



- TD-FARL on ConnectFour, 10 runs.
- TD-FARL against the perfect-playing agents AlphaBeta (AB) and AlphaBeta-with-distant-losses (AB-DL).

- The ideal win rates against AB and AB-DL are 0.5.
- Solid curves: TD-FARL.
- Remarkable: TD-FARL learns *solely from self-play* to defeat AB and AB-DL
- Dashed curves: when FARL is turned off.

Results Various Games

Game	N	evaluated vs.	win rates or scores		other RL research
			FARL	no-FARL	
2048	1		**142 000** ± 1 000	122 000 ± 900	80 000
TicTacToe	2	Max-N_{10}	**49%** ± 5%	18% ± 6%	[Jaśkowski, 2018]
ConnectFour	2	AB	**49.5%** ± 0.5%	3.5% ± 0.1%	0.0% ± 0.0%
		AB-DL	**46.5%** ± 0.5%	0.0% ± 0.1%	[Dawson, 2020]
Hex 6x6	2	$MCTS_{10000}$	**81%** ± 5%	0.0% ± 0.2%	
Othello	2	$Edax_{d1}$	**55%** ± 1%	53% ± 1%	
		Bench	95% ± 0.3%	**96%** ± 0.2%	87.1% ± 0.9%
Nim 3x5	2	Max-N_{15}	**50%** ± 1%	12% ± 6%	[van der Ree, 2013]
Nim3P 3x5	3	Max-N_{15}	**0.33** ± 0.03	0.03 ± 0.01	
		$MCTS_{5000}$	**0.78** ± 0.02	0.09 ± 0.02	

- $MCTS_N$: Monte-Carlo Tree Search with N iterations [Browne et al., 2012]
- Max-N_d: Max-N tree search with depth d [Korf, 1991]
- $Edax_{d1}$: Edax Othello agent with depth 1 [Delorme, 2019]

Discussion of Results

Findings:

- TD-FARL beats MCTS on several games.
- Many of the opponents (AB, AB-DL, Max-N$_d$) are perfect players. A win rate of 50% means perfect play for TD-FARL.
- For all games: No success if we switch off FARL.
- The one exception: Othello. – Why? Still open.
- TD-FARL is a versatile RL-algorithm on many games. Where comparable, it seems stronger than other general RL agents.

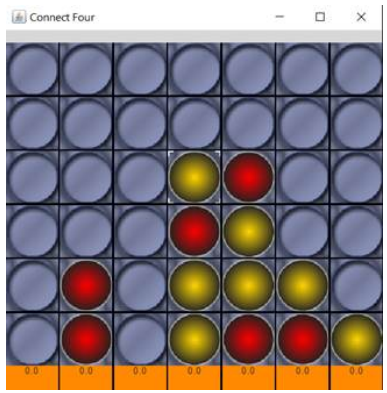
Computatation Times

Agent	Game	Moves/second during ...	
		... game learning	... game play
TD-FARL	2048	66.000	94.000
MCTSE (1000 iter)	2048	–	120
TD-FARL	Connect-4	7.900	40.400
MCTS (1000 iter)	Connect-4	–	54
TD-FARL	Hex 5x5	17.600	20.500
MCTSE (1000 iter)	Hex 5x5	–	31

- The above numbers are for a standard single-core CPU, no GPU.
- TD-FARL is relatively fast, compared to MCTS, GGP or other deep learning agents.
- Training a TD-FARL agent on Connect-4 completes in **1 hour**.

Conclusion

- Game learning is a fascinating & challenging optimization area
- We presented a unifying, versatile game learning algorithm **TD-FARL** which works for $N = 1, 2, 3, \ldots$ players
- On the games tested so far: Often better than other RL approaches
- Relatively fast learning on standard hardware
- TD-FARL is part of the open-source GBG framework: https://github.com/WolfgangKonen/GBG [Konen, 2020]



Future Work

Planned work in the open-source **GBG framework** for game learning (https://github.com/WolfgangKonen/GBG):

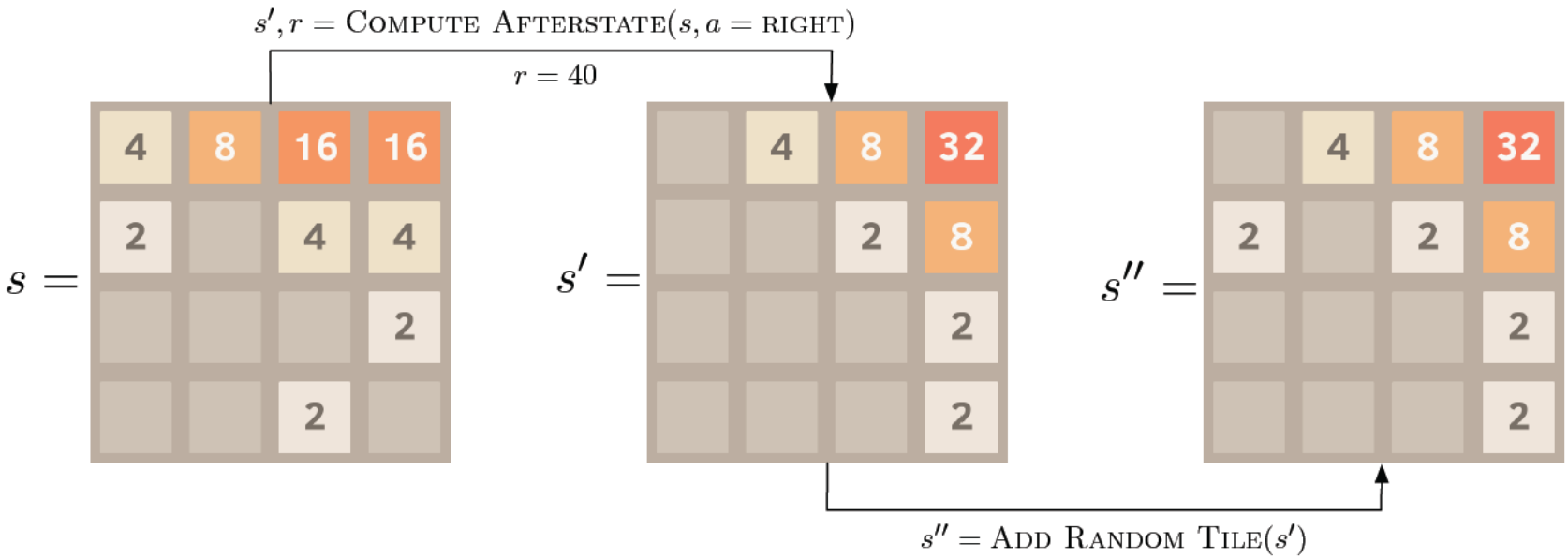
- More games to test TD-FARL on
  - Poker, Blackjack (and other non-trivial N-player games)
  - Rubik's Cube (2x2x2 and 3x3x3)
  - ...
- AlphaGo-inspired [Silver et al., 2017] agents for 'small budget'
- ...

The GBG framework evolves through student projects and other researchers' contributions.

Thank You For Your Attention!

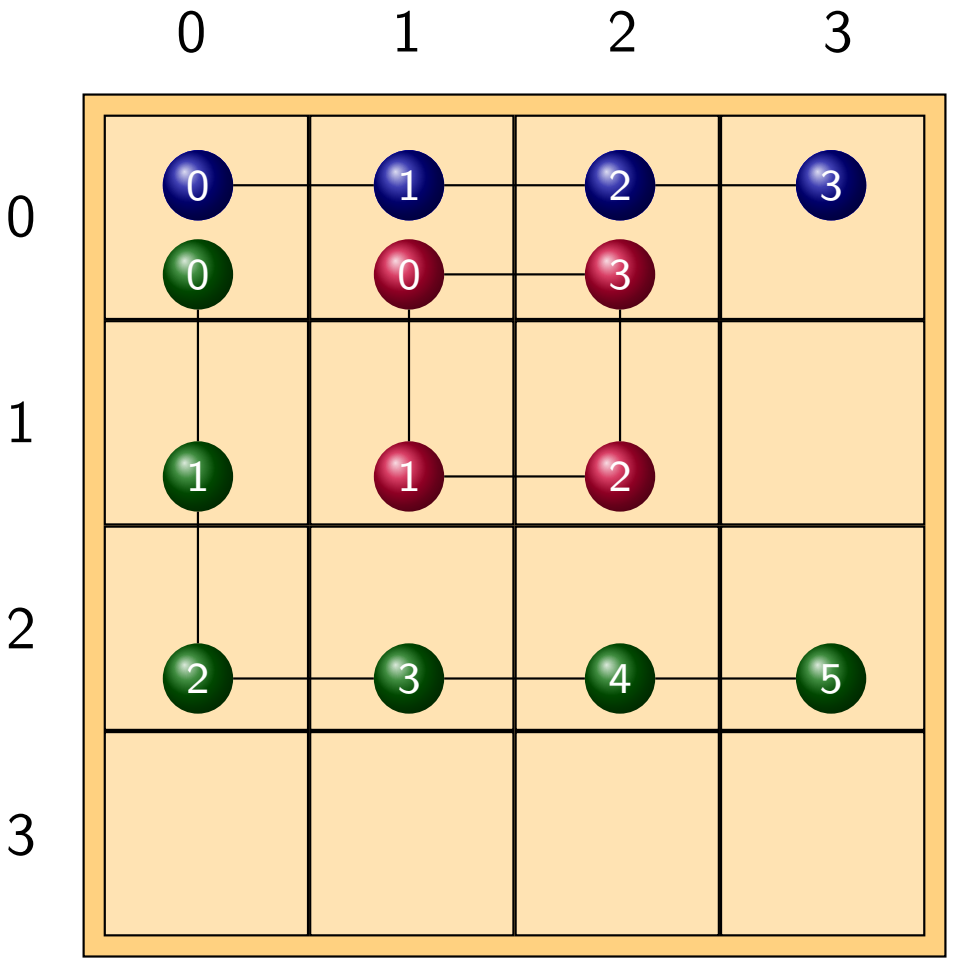
Questions?

Side Remark: Afterstates



- Aftestate: The state after the agent has acted, before the environment has altered the state.
- Afterstates help to reduce complexity: since there are less afterstates than next states.

Side Remark: N-Tuples



Two 4-tuples and one 6-tuple

- **N-tuples**: High-dimensional feature space over board cells [Lucas, 2008]
- m n-tuples, P positional values
- $x_{\nu,i} = 1$ if feature i of n-tuple ν is present on board
- Learnable lookup-table (weights) $w_{\nu,i}$
- Value estimation

$$V^{(est)} = \sigma\left(\sum_{\nu=1}^{m}\sum_{i=0}^{P^n-1} x_{\nu,i} w_{\nu,i}\right)$$

Further Reading I

[Browne et al., 2012] Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. (2012).
A survey of Monte Carlo tree search methods.
IEEE Transactions on Computational Intelligence and AI in Games, 4(1):1–43.

[Dawson, 2020] Dawson, R. (2020).
Learning to play Connect-4 with deep reinforcement learning.
https://codebox.net/pages/connect4, retrieved August,21,2020.

[Delorme, 2019] Delorme, R. (2019).
Edax, version 4.4.
https://github.com/abulmo/edax-reversi, retrieved August,1,2020.

Further Reading II

[Jaśkowski, 2018] Jaśkowski, W. (2018).
Mastering 2048 with delayed temporal coherence learning, multistage weight promotion, redundant encoding, and carousel shaping.
IEEE Transactions on Games, 10(1):3–14.

[Konen, 2020] Konen, W. (2020).
The GBG class interface tutorial V2.2: General board game playing and learning.
Technical report, Research Center CIOP (Computational Intelligence, Optimization and Data Mining), Oct.

[Korf, 1991] Korf, R. E. (1991).
Multi-player alpha-beta pruning.
Artificial Intelligence, 48(1):99–111.

Further Reading III

[Lucas, 2008] Lucas, S. M. (2008).
Learning to play Othello with n-tuple systems.
Australian Journal of Intelligent Information Processing, 4:1–20.

[Silver et al., 2017] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017).
Mastering the game of Go without human knowledge.
Nature, 550(7676):354–359.

[van der Ree, 2013] van der Ree, M. (2013).
Reinforcement learning in the game of Othello: Learning against a fixed opponent and learning from self-play.
In *Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 108–115.